Оптимизация алгоритма сбалансированного разбиения графа

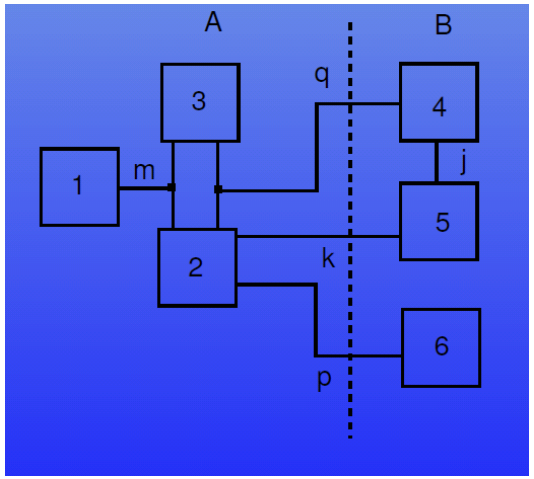
Русаков А.С., Шеблаев М.В.

ИППМ РАН, eASIC

Постановка задачи

- *H(V, E)* – гиперграф, *V* — вершины, *E* – гиперребра,
- $w(e_i)$, $w(v_i)$ - веса гиперребер и вершин.
- Требуется найти множества вершин A и B такие, что:
  -
  - Для { | стоимость разреза

$$Cut = \sum_{e_i \in \Psi} w(e_i).$$



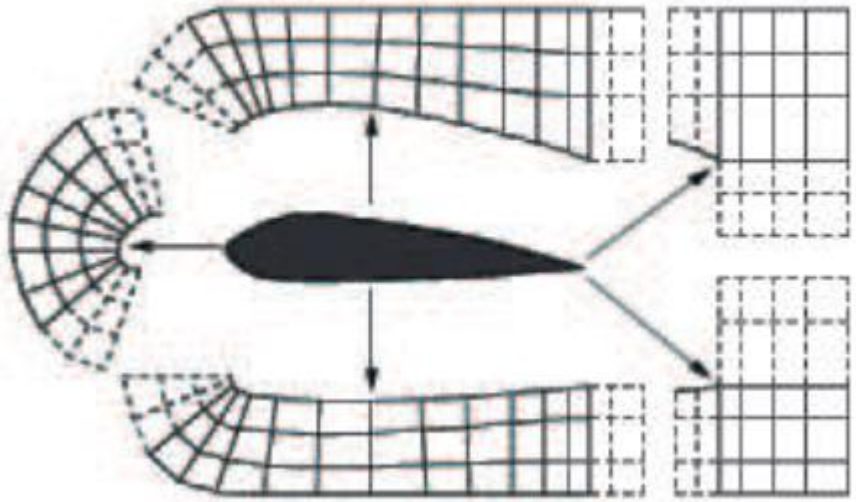
  - Выполняется условие баланса

$$1 - \varepsilon \le \sum_{v_i \in A} w(v_i) \Big/ \sum_{v_i \in B} w(v_i) \le 1 + \varepsilon$$

Применение в HPC

- Многопроцесорная обработка нерегулярных сеток

- Балансировка загрузки ресурсов при минимизации межпроцессорной коммуникации



Применение

- Задачи линейной алгебры
- Задачи дискретной математики
- Задачи САПР СБИС

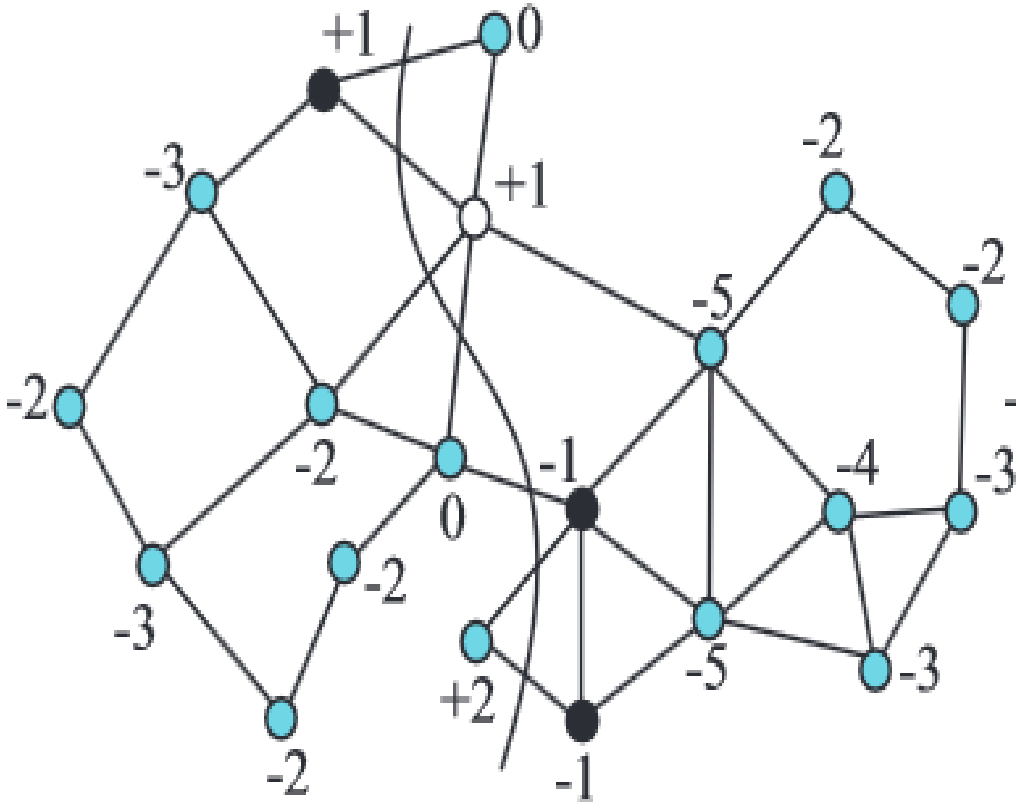
- Известные пакеты: MLPart, hMetis, ZOLTAN, PaToH, SCOTCH

Методы решения

- Комбинаторные методы: итерационная бисекция, ветви и границы
- Спектральные методы: анализ матрицы Лапласа и ее собственные вектора
- Имитационное моделирование: генетические алгоритмы, имитация отжига, и т.д.
- Пошаговые эвристики: Kernighan-Lin и Fiduccia-Mattheyses

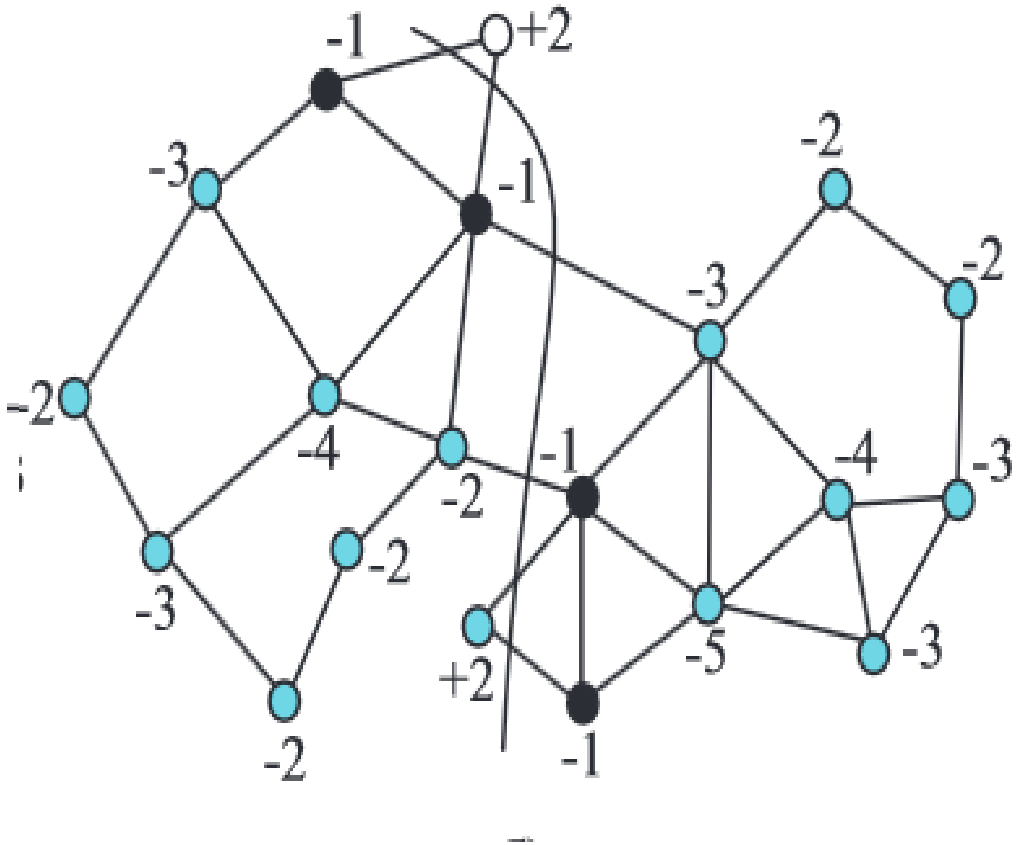
FM-алгоритм

- Начальное разбиение
- FS(v) - перетягивающие силы
- TE(v) – удерживающие силы
- = FS(V) – TE(V) -- стоимость перемещения



FM-алгоритм

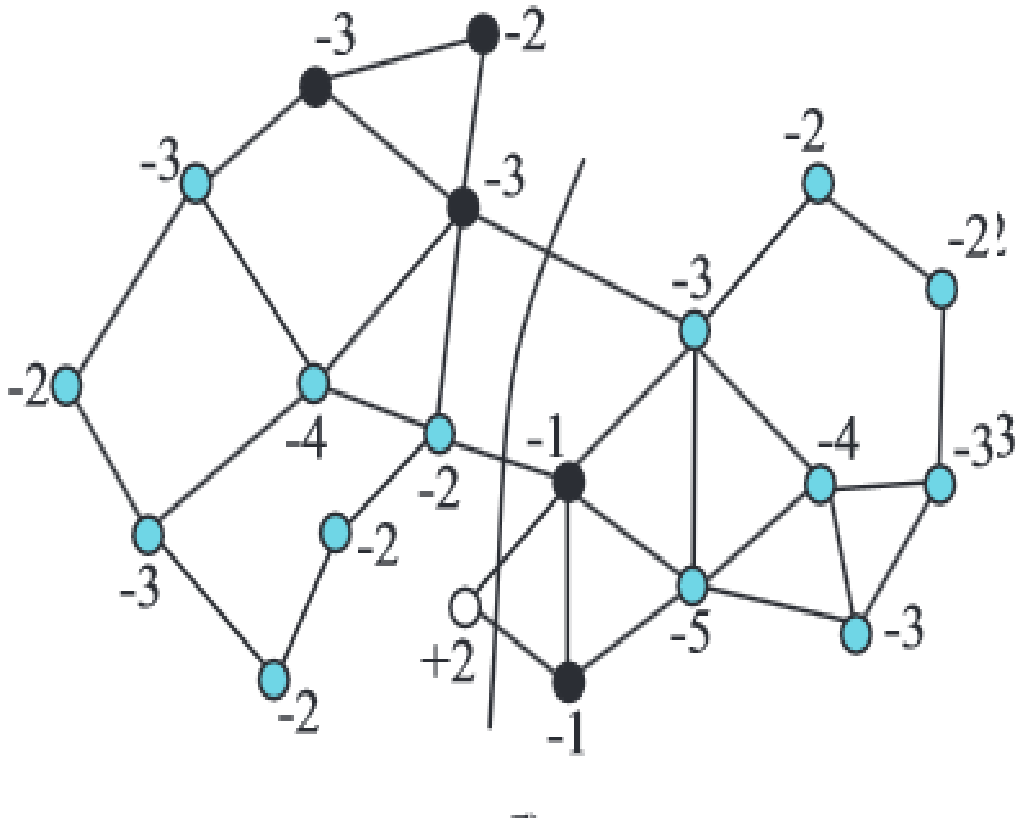
- Пошаговое перемещение базовой вершины
- Проверка условий баланса



FM-алгоритм

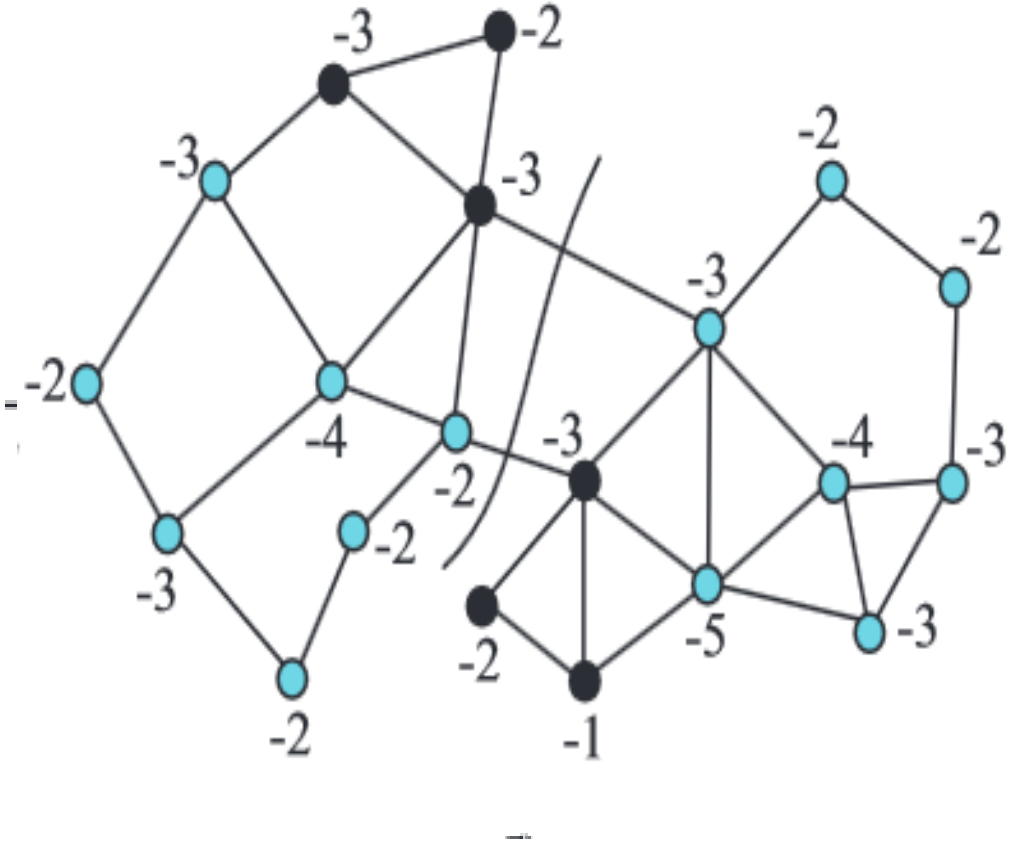
- Пошаговое перемещение базовой вершины

- Проверка условий баланса

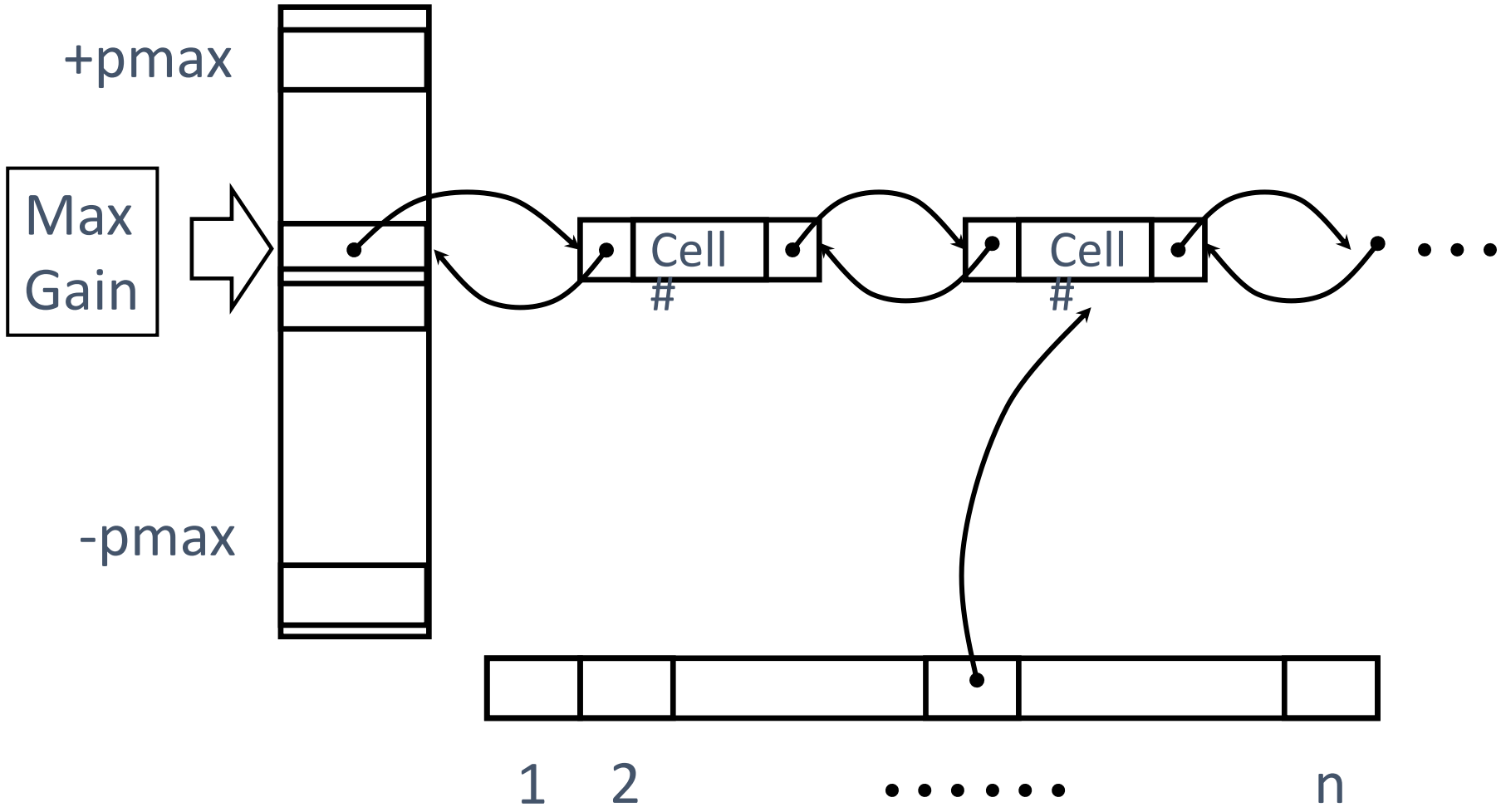


FM-алгоритм

- Зависимость от начального разбиения
- Разрешение коллизий при выборе базовой вершины



Gain Bucket Data Structure



FM-алгоритм: gain bucket

- Classic bucket

```
*cell = max_gain_cell
while( cell != NULL) {
    if ( IsBalancedMove(cell)    return cell;
    Remove cell from bucket(cell)
    cell = cell->next
}
return NULL
```

O(1)

- Bucket with restart

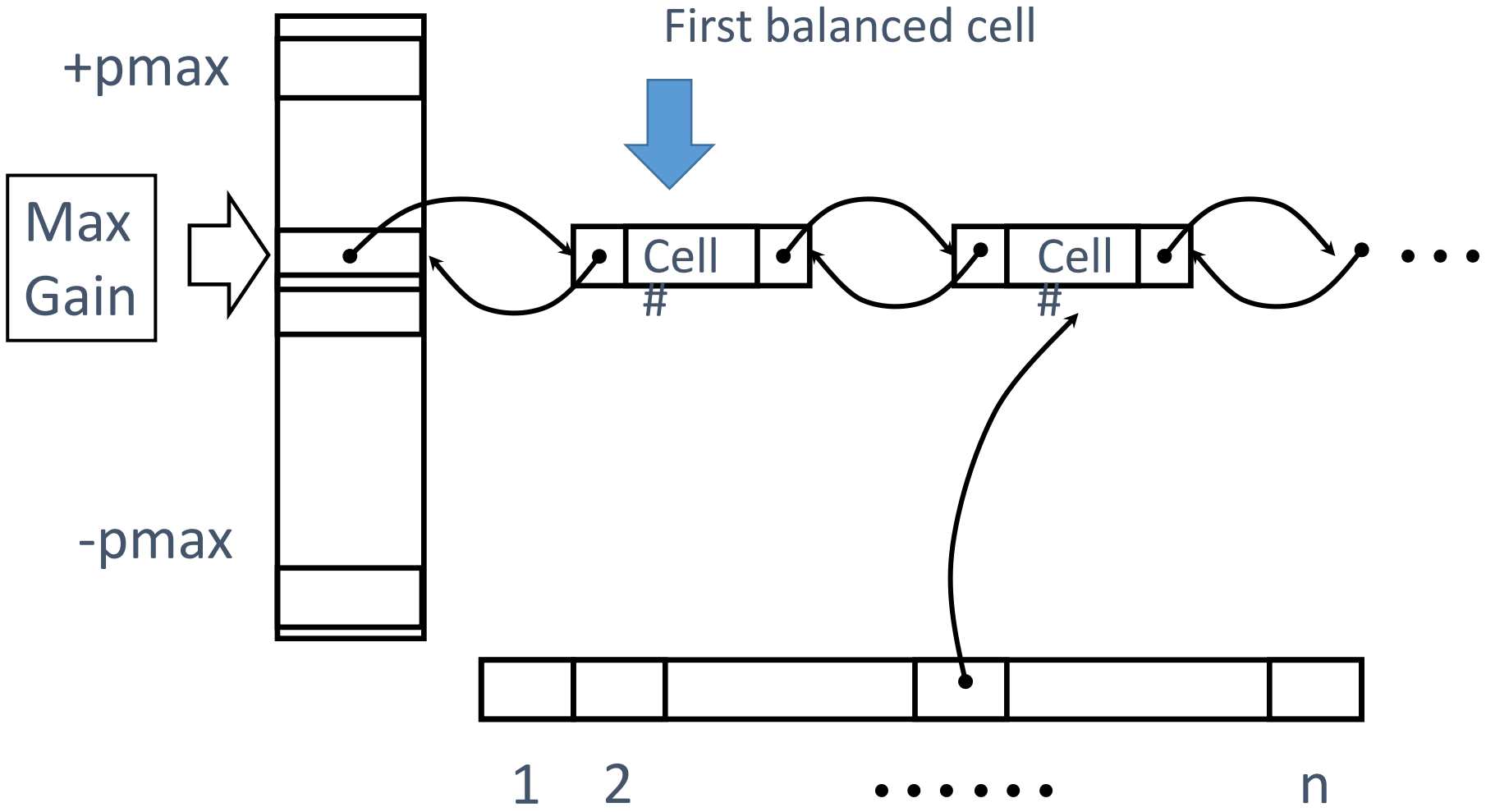
```
*cell = max_gain_cell
while( cell != NULL) {
    if ( IsBalancedMove(cell)    return cell;
    cell = cell->next
}
return cell;
```

O(n), лучшее качество разреза

FLAT FM. 2% imbalance

	bucket with restart			Classic bucket			new bucket		
TestName	Cut	Time	Quality	cut	time	Quality	cut	time	Quality
ibm01:	303	1	1,00	328	1	0,92	393	2	0,77
ibm02	446	4	0,84	768	2	0,49	373	3	1,00
ibm03:	1734	4	0,85	2513	3	0,59	1558	3	0,95
ibm03	1042	7	0,94	1423	4	0,69	1093	5	0,90
ibm05:	2010	7	0,93	3201	4	0,59	1878	5	1,00
ibm06:	1076	9	0,96	2752	4	0,38	1173	7	0,88
ibm07:	1505	16	0,89	1574	9	0,85	1332	12	1,00
ibm08:	1937	35	1,00	5242	10	0,37	2491	17	0,78
ibm09:	1246	20	1,00	3806	12	0,33	1457	15	0,86
ibm10:	2638	49	0,85	4204	17	0,53	2230	24	1,00
ibm11:	2612	32	0,82	5757	18	0,37	2136	22	1,00
ibm12:	3722	73	1,00	8350	18	0,45	4741	30	0,79
ibm13	2188	39	0,76	3042	22	0,54	2036	28	0,81
ibm14:	3855	69	0,82	7528	43	0,42	4561	54	0,70
ibm15:	4948	168	0,87	9542	51	0,45	5246	75	0,82
ibm16:	3683	179	1,00	10828	57	0,34	4356	82	0,85
ibm17:	3285	117	1,00	13236	60	0,25	5434	79	0,60
ibm18:	3345	191	1,00	9522	63	0,35	3758	102	0,89
Score		1020,00	16,52		398,00	8,89		565,00	15,58

Gain Bucket Data Structure



FM-алгоритм: first balanced cell

- Our modification of bucket. Introduce a new pointer to first balanced cell:

```
GetBestMove() {
    *cell = max_gain_cell
    If (first_balanced) cell = first_balanced;
    while( cell != NULL) {
        if ( IsBalancedMove(cell) {
            first_balanced = cell->next;
            return cell;
        }
        cell = cell->next
    }
    return NULL
}
```

Starts search from the “good” cell. Good tradeoff of runtime and cut size

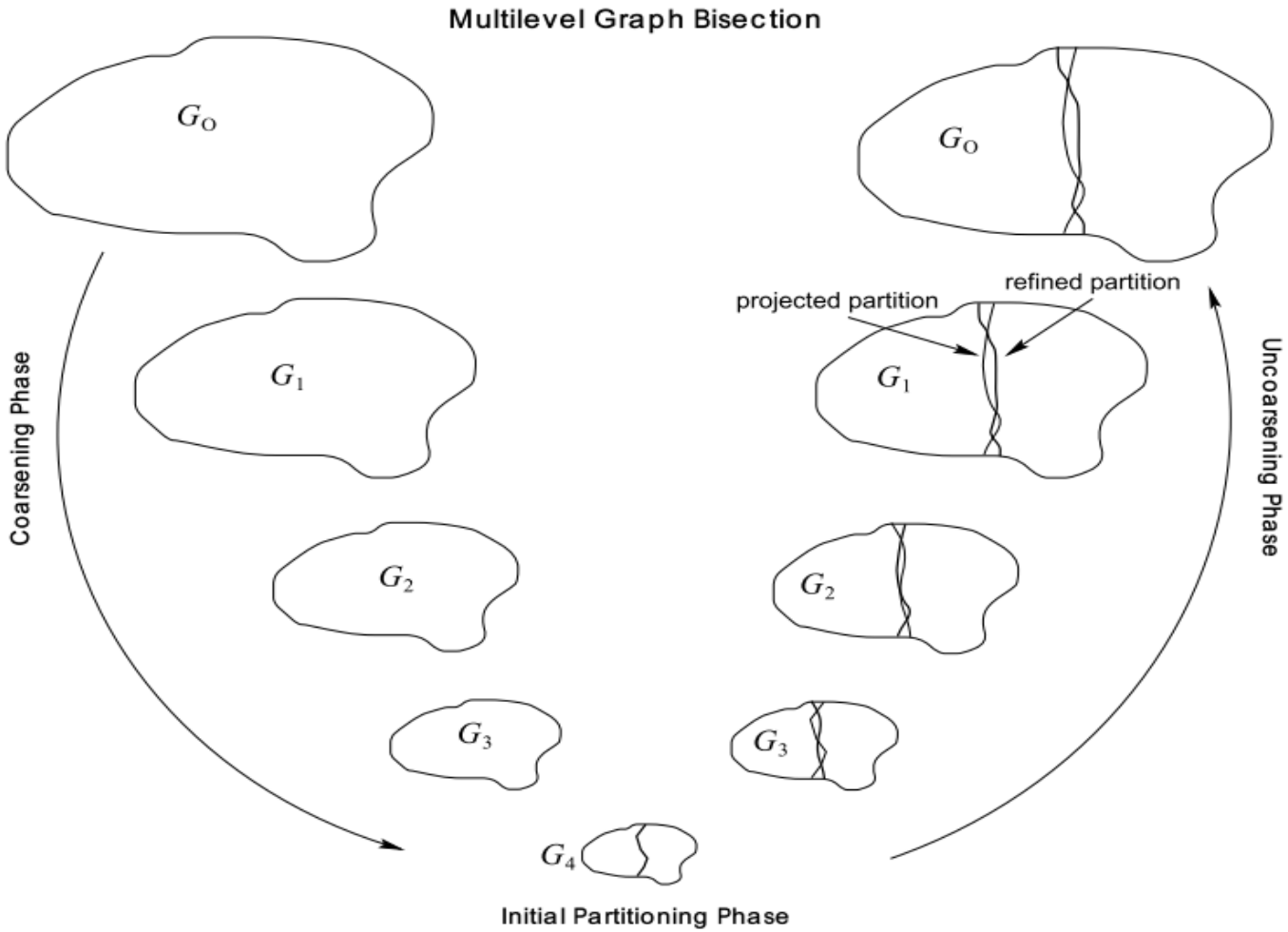
- Update of the gain cell. Reset

```
UpdateGainCell(new gain) {
  ...
  If(new_gain > first_balanced_gain)  first_balance = NULL
  ...
}
```

FLAT FM. 2% imbalance

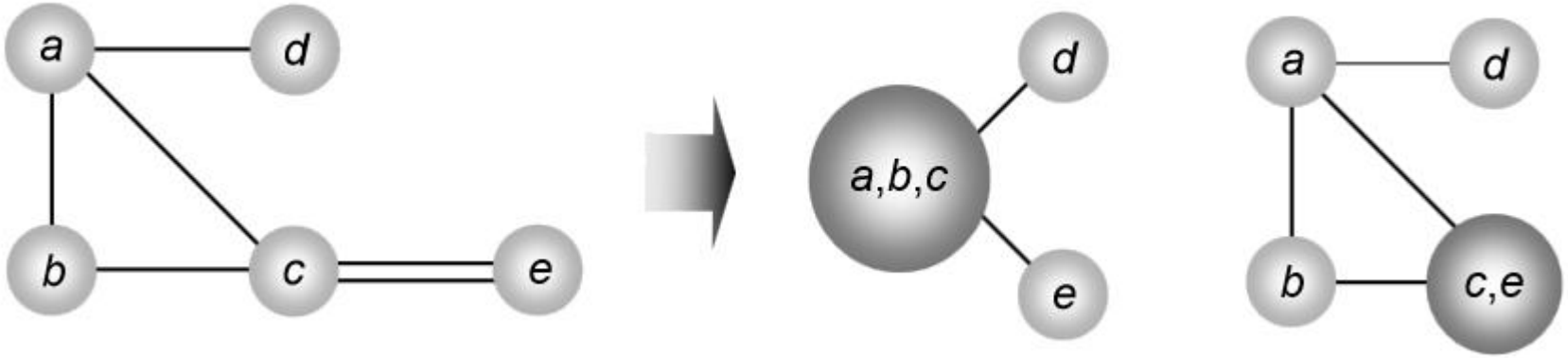
	bucket with restart			Classic bucket			new bucket		
TestName	Cut	Time	Quality	cut	time	Quality	cut	time	Quality
ibm01:	303	1	1,00	328	1	0,92	393	2	0,77
ibm02	446	4	0,84	768	2	0,49	373	3	1,00
ibm03:	1734	4	0,85	2513	3	0,59	1558	3	0,95
ibm03	1042	7	0,94	1423	4	0,69	1093	5	0,90
ibm05:	2010	7	0,93	3201	4	0,59	1878	5	1,00
ibm06:	1076	9	0,96	2752	4	0,38	1173	7	0,88
ibm07:	1505	16	0,89	1574	9	0,85	1332	12	1,00
ibm08:	1937	35	1,00	5242	10	0,37	2491	17	0,78
ibm09:	1246	20	1,00	3806	12	0,33	1457	15	0,86
ibm10:	2638	49	0,85	4204	17	0,53	2230	24	1,00
ibm11:	2612	32	0,82	5757	18	0,37	2136	22	1,00
ibm12:	3722	73	1,00	8350	18	0,45	4741	30	0,79
ibm13	2188	39	0,76	3042	22	0,54	2036	28	0,81
ibm14:	3855	69	0,82	7528	43	0,42	4561	54	0,70
ibm15:	4948	168	0,87	9542	51	0,45	5246	75	0,82
ibm16:	3683	179	1,00	10828	57	0,34	4356	82	0,85
ibm17:	3285	117	1,00	13236	60	0,25	5434	79	0,60
ibm18:	3345	191	1,00	9522	63	0,35	3758	102	0,89
Score		1020,00	16,52		398,00	8,89		565,00	15,58

Иерархический подход



Иерархический подход

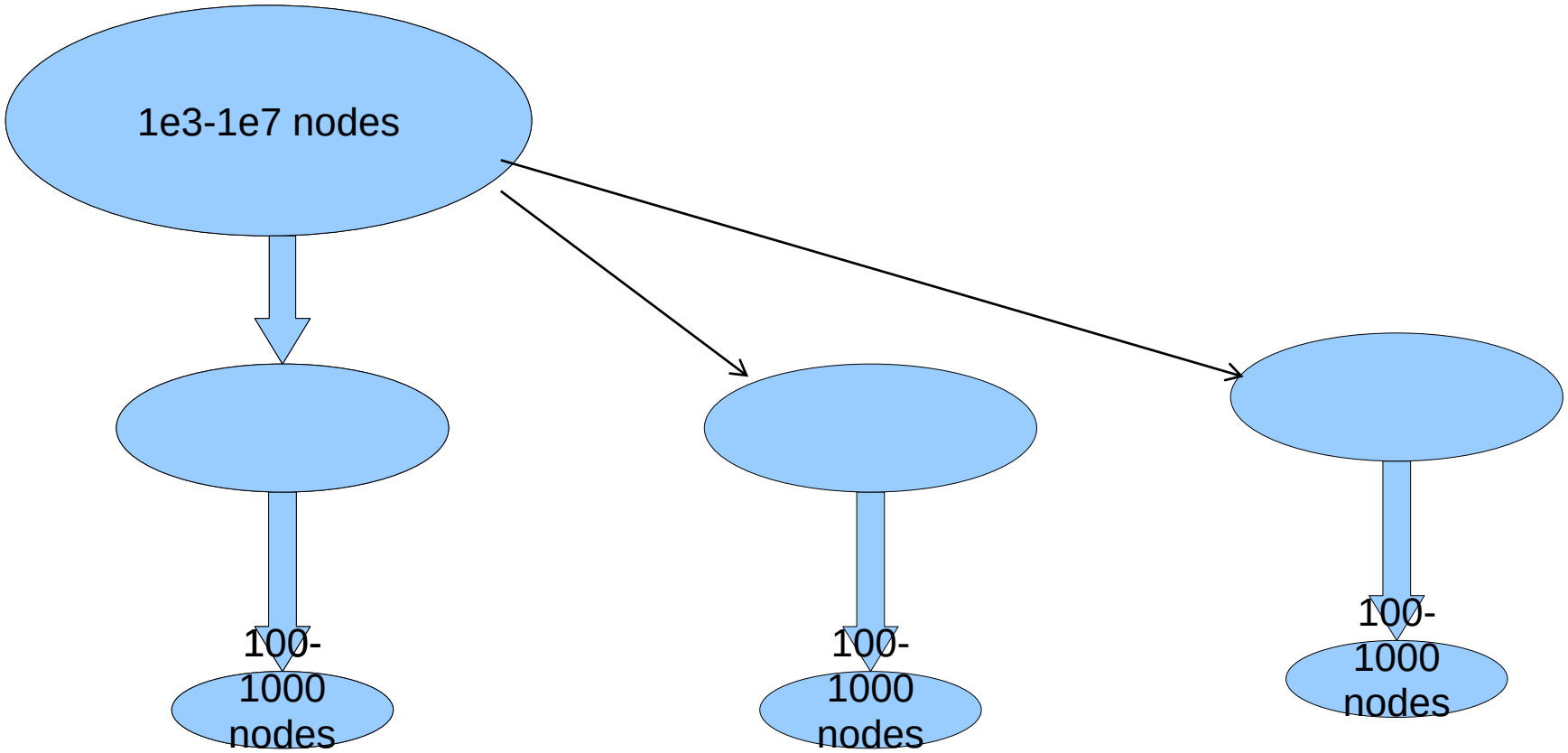
- Разные способы кластеризации



- FC, HEC, HEM, MHEC, PinHEM. Выбор наилучшей кластеризации

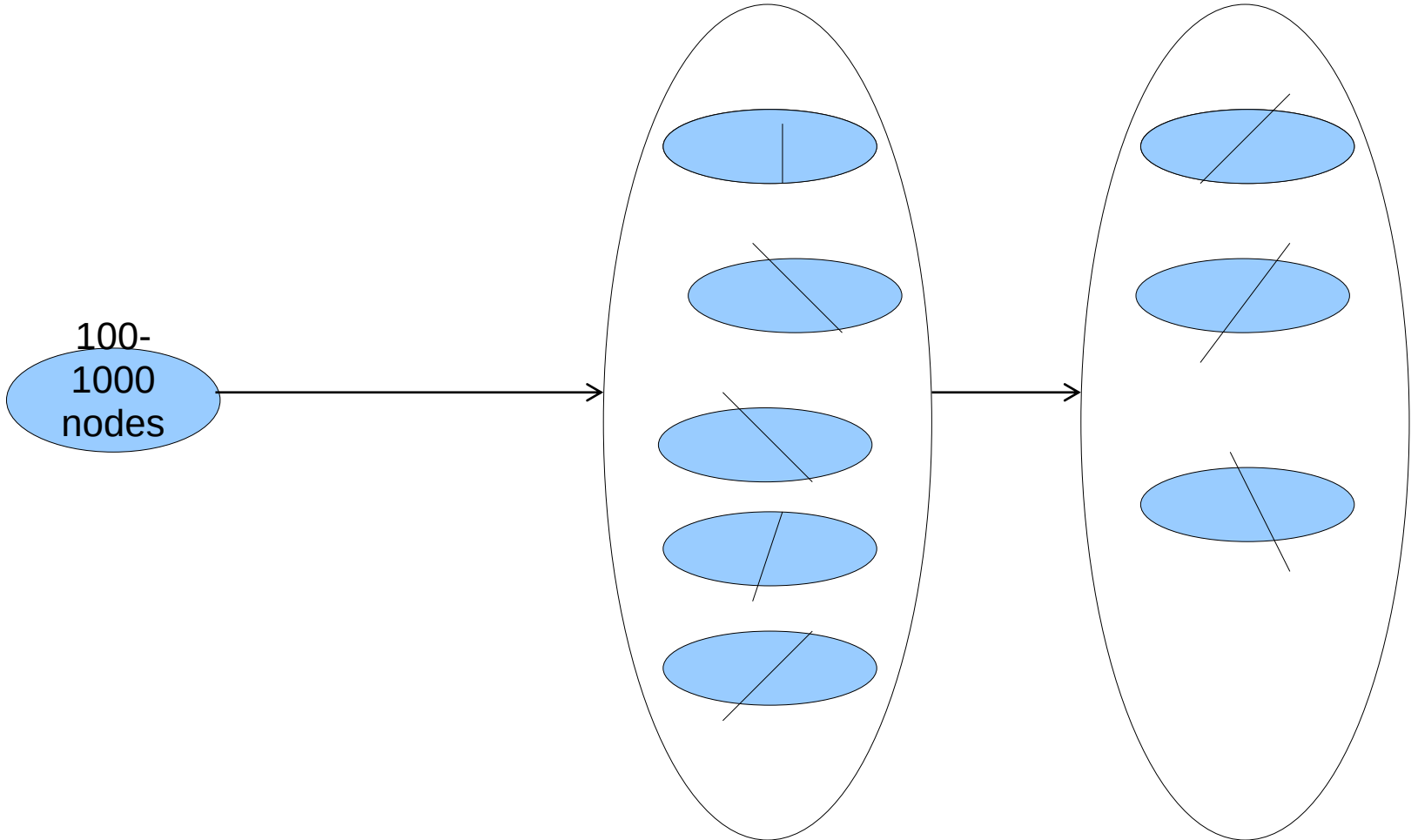
Многоуровневый алгоритм

Этап 1. Кластеризация или свертывание входного графа. Уменьшение размерности до нескольких сотен узлов. Одновременно используются FC, FCPIN, HEM, HEC, MHEC



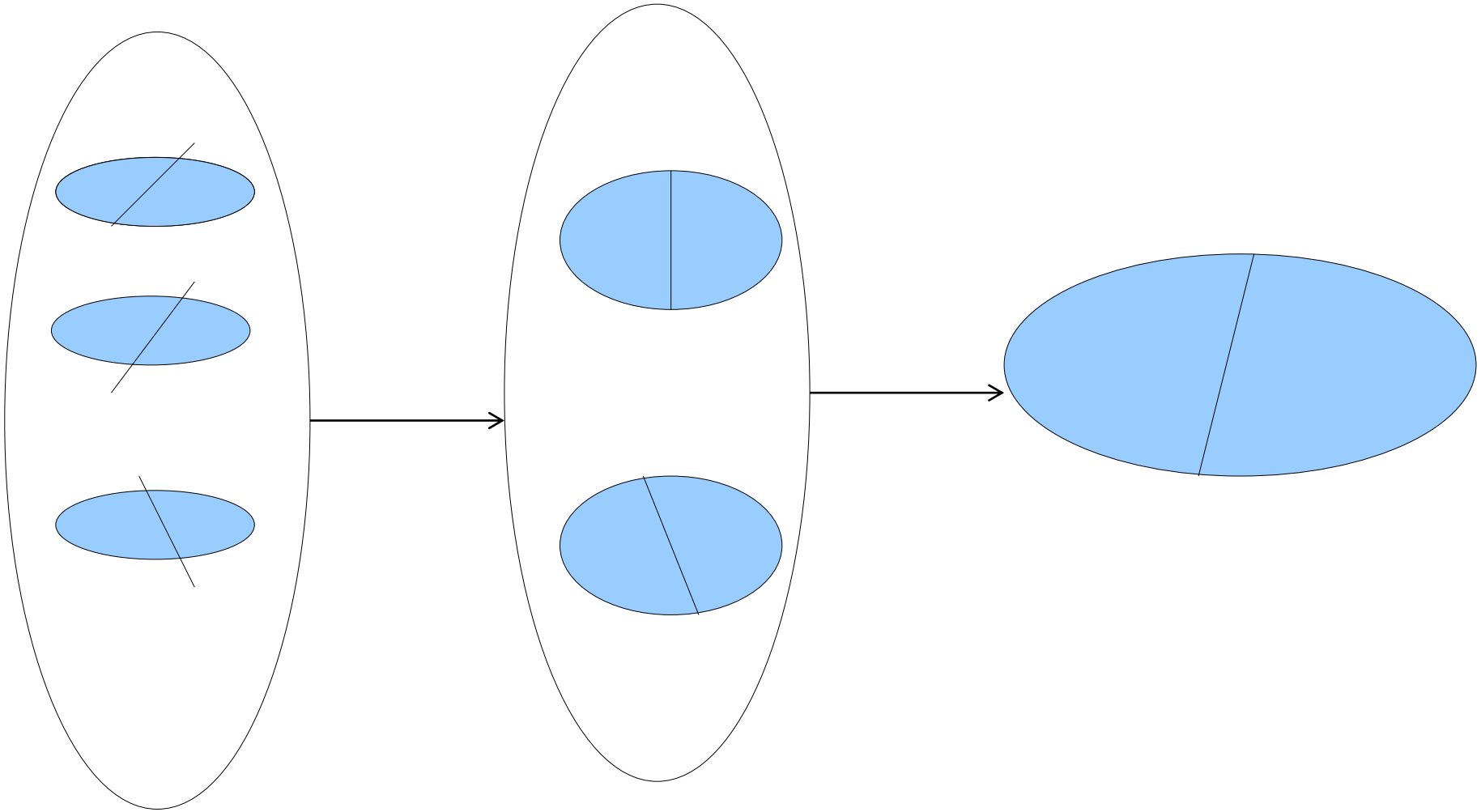
Многоуровневый алгоритм

Этап 2. Первоначальное разбиение кластеризованного графа
K разных решений. Выбор N лучших решений для улучшения



Многоуровневый алгоритм

Этап 3. Последовательно уточнение разбиений.
Постепенное отбрасование плохих решений

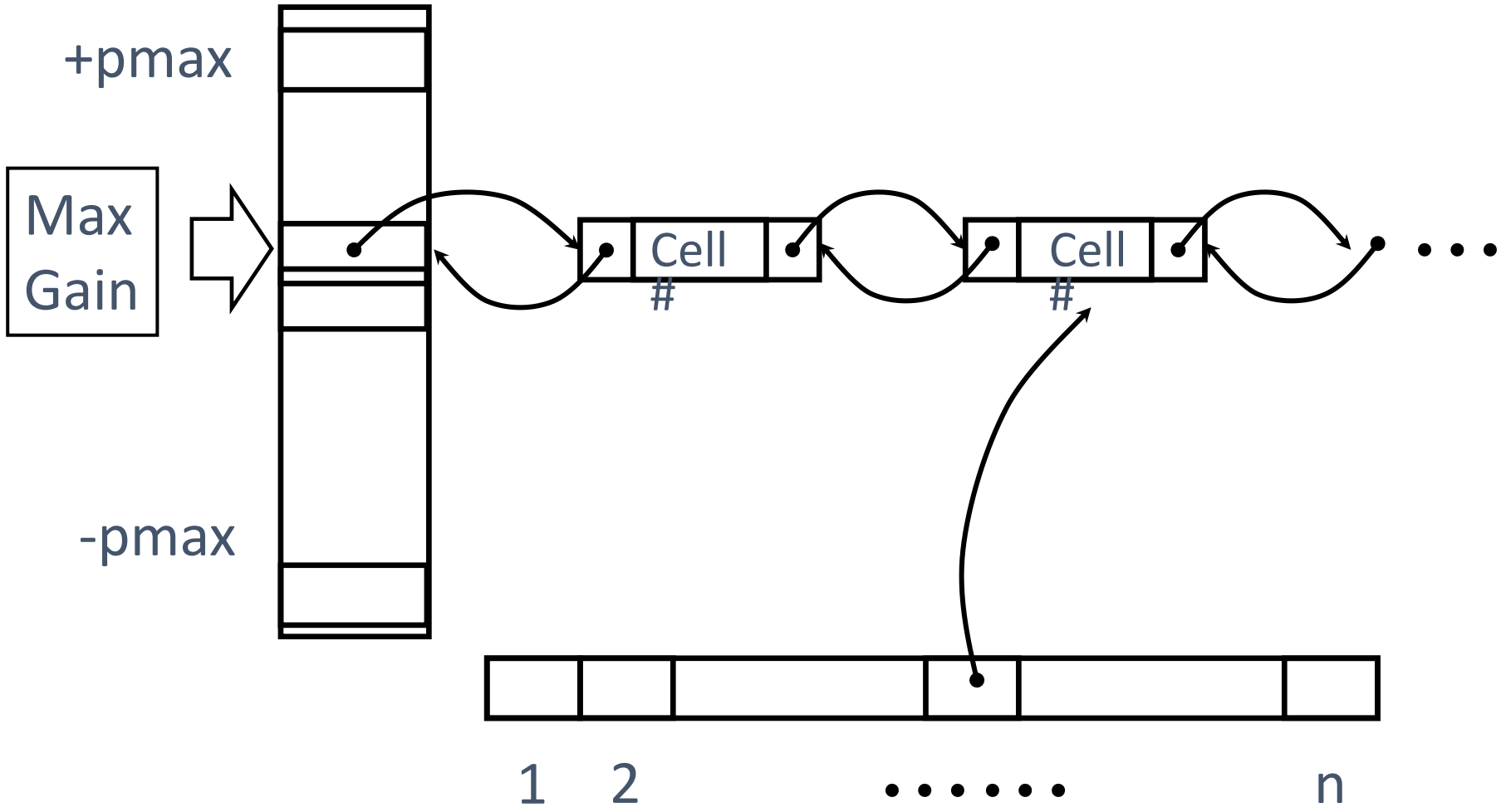


Бенчмарки

	Bucket with restarts			New bucket			Classic bucket		
TestName	Cut	Time	Score	Cut	Time	Score	Cut	Time	Score
ibm01:	219	4	1,00	245	4	0,89	336	3	0,65
ibm02	269	5	0,99	269	5	0,99	266	5	1,00
ibm03:	751	10	1,00	791	8	0,95	911	7	0,82
ibm03	535	9	0,96	515	8	1,00	513	7	1,00
ibm05:	1733	11	1,00	1732	12	1,00	1725	10	1,00
ibm06:	585	11	0,89	535	11	0,98	788	11	0,66
ibm07:	802	22	1,00	833	17	0,96	820	15	0,98
ibm08:	1190	32	1,00	1190	17	1,00	1187	16	1,00
ibm09:	533	18	1,00	535	17	1,00	535	16	1,00
ibm10:	1119	39	0,97	1084	17	1,00	1216	17	0,89
ibm11:	874	34	0,97	868	26	0,97	844	27	1,00
ibm12:	2195	64	0,95	2185	33	0,95	2084	26	1,00
ibm13	1073	62	0,95	1111	43	0,92	1024	43	1,00
ibm14:	1938	100	1,00	1955	77	0,99	1960	74	0,99
ibm15:	2567	200	1,00	2653	92	0,97	2709	93	0,95
ibm16:	1788	151	1,00	1805	92	0,99	1868	104	0,96
ibm17:	2296	150	1,00	2362	158	0,97	2648	92	0,87
ibm18:	1943	49	1,00	2232	40	0,87	2057	23	0,94
Score		971,00	17,67		677,00	17,40		589,00	16,71

Achieve almost the same cut size with 30% speeding up

Gain Bucket Data Structure



LIFO*

- Стратегии выбора перемещения вершины
  - FIFO
  - LIFO
  - RANDOM
  - LIFO* - «переборчивый LIFO»
    - Улучшение стоимости перемещения – LIFO
    - Ухудшение стоимости перемещения - FIFO

LIFO + LIFO*

test case	LIFO	LIFO*		LIFO+LIFO*	
ibm01	262	252	3,97%	245	6,49%
ibm02	269	346	-22,25%	269	0,00%
ibm03	826	850	-2,82%	822	0,48%
ibm04	537	539	-0,37%	537	0,00%
ibm05	1731	1726	0,29%	1731	0,00%
ibm06	596	718	-16,99%	596	0,00%
ibm07	820	820	0,00%	820	0,00%
ibm08	1190	1190	0,00%	1190	0,00%
ibm09	535	535	0,00%	535	0,00%
ibm10	1088	1011	7,62%	1088	0,00%
ibm11	877	820	6,95%	877	0,00%
ibm12	2185	2193	-0,36%	2185	0,00%
ibm13	1170	1175	-0,43%	1166	0,34%
ibm14	1955	1969	-0,71%	1955	0,00%
ibm15	2488	2639	-5,72%	2488	0,00%
ibm16	1805	1824	-1,04%	1789	0,89%
ibm17	2378	2403	-1,04%	2378	0,00%
ibm18	1959	1931	1,45%	1959	0,00%
			-31,47%		8,20%

State-of-Art и бенчмарки

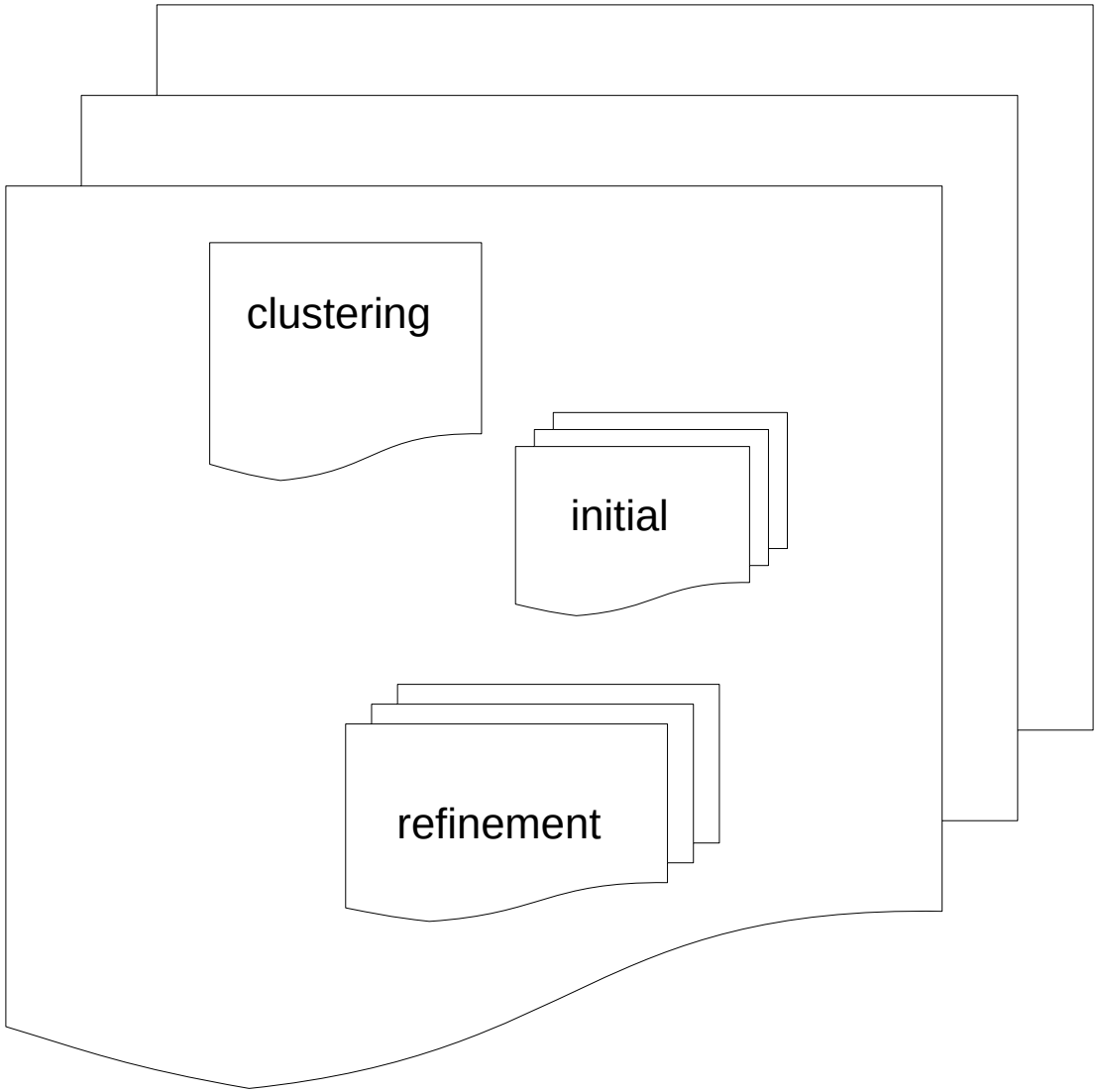
test case	hMETIS	MLPart	eASIC
ibm01	245	227	**233**
ibm02	299	294	**269**
ibm03	855	818	**763**
ibm04	534	531	**502**
ibm05	1741	1750	**1739**
ibm06	605	564	**525**
ibm07	794	**793**	839
ibm08	1208	1206	**1168**
ibm09	**524**	527	546
ibm10	1193	1211	**1119**
ibm11	**813**	842	821
ibm12	2131	2353	**2089**
ibm13	931	**1036**	1050
ibm14	1865	1860	**1792**
ibm15	**2221**	2243	2494
ibm16	**1713**	1853	1739
ibm17	2460	**2353**	2376
ibm18	**1706**	1737	1776

Выводы

- Классический FM-алгоритм применим для многих практических задач
- Корзина движений с перезапуском: улучшение качества без существенной потери времени
- Многоуровневый подход: оптимизация времени
- Предложены новые модификации известных алгоритмов

Резервные слайды

Параллельный многоуровневый алгоритм



Независимые задачи для параллельного вычисления:

1) решение с выбранной схемой кластеризации. 5-6 задач
2) первоначальное разбиение графа. K решений. K = 20-100
3) Последовательное улучшение N задач. N = 20-50